Quantum Sequencer for the Minimal Test Synthesis of Black-box Functionality

Vladimir Hahanov
Kharkov National University of Radioelectronics, Ukraine
hahanov@icloud.com

Igor Iemelianov
Kharkov National University of Radioelectronics, Ukraine
apot@kture.kharkov.ua

Svetlana Chumachenko
Kharkov National University of Radioelectronics, Ukraine
svetachumachenko@icloud.com

Ivan Hahanov
Kharkov National University of Radioelectronics, Ukraine
ivanhahanov@icloud.com

Irina Hahanova
Kharkov National University of Radioelectronics, Ukraine
hahanova@mail.ru

Abstract

Quantum memory-driven computing on the classical computers for design and test of black-box functionality is considered. A method for synthesis and minimization test for the black-box functionality, based on a qubit derivative matrix and sequencer for searching a quasi-optimum coverage, is proposed. Examples of quantum memory-driven design and test minimization of the Schneider logic circuit are presented. An architecture and algorithm for parallel search of a quasi-minimal set of test vectors based on the logical structure is developed. The technological advantages of the qubit coverage leverage for increasing the speed of performance due to the parallel solution of the test synthesis and analysis for single stuck-at-faults of external and internal variables are shown.

1. Introduction

Quantum computing is currently the most important area of research, which is engaged by all the leading companies on the planet related to the IT industry (d-wave, IBM, Google, NASA, Acronis). Naturally, the activity of these companies is primarily aimed at creating a prototype of a quantum computer of the future, capable of solving NP-complete tasks within an acceptable time. It is understandable that academic scientists are increasingly engaged in research in the field of creating new mathematical theories, methods and algorithms oriented to the parallel solution of actual problems in the metrics of quantum computers. Otherwise, there may be a situation when naked quantum computers appear on the market, for which there will be no software or cloud applications. A vivid example of the quantum-oriented problems is the search for optimal test coverage for all stuck-at-faults for digital system on chip. For more than 100 years, the Cantor algebra has existed in discrete mathematics and Hasse's isomorphic structure, which is a model of quantum processing based on the properties of superposition and entanglement for the efficient, and parallel solution of combinatorial problems [1-3].

2. Quantum Hasse Sequencer of the Quasi-Optimal Coverage

The quantum computing architecture is proposed for a significant increase in the performance to solve discrete optimization problems [1-5] in the field Design and Test. Hardware-oriented models of parallel taking the Boolean (set of all subsets) for the universum of n-primitives are described. Hasse-sequencer [4-5] is focused on solving the problems of coverage, minimization tests, logic functions, data compression, synthesis, and analysis of digital systems.

The purpose of creation the Hasse sequencer is to significantly reduce the time for solving optimization problems by parallel computing vector logical operations over the set of all subsets of primitive components by increasing the memory for storing matrix data.

The objectives: 1) Definition of data structures for taking a Boolean when solving the problem of coverage columns of a matrix $M = |M_{ij}|, i = \overline{1,m}; j = \overline{1,n}$ by 1-unit row values. In particular, for m = n = 8, it is necessary to execute in parallel a logical operation over 256 variants of all possible combinations of the matrix rows that make up the Boolean. The sequencer instruction system includes logical operations (and, or, xor) over vectors with m-lenth. 2) Development of Hasse sequencer architecture for parallel computation of $2^n - 1$ combination's variants aimed at the optimal solution of the NP-complete coverage problem. 3) Implementation of the Hasse sequencer prototype on programmable logic and verification of the hardware solution by examples of minimizing logic functions.

As an example, it is proposed to solve the problem of search the optimal 1-unit coverage of all columns by the minimum number of rows of the matrix M:

M	1	2	3	4	5	6	7	8
a	1	.	.	.	.	1	.	.
b	.	.	1	.	.	.	1	.
c	1	.	.	.	1	.	1	.
d	.	1	.	1	.	.	.	1
e	.	1	.	.	1	.	.	.
f	1	.	1	.	.	1	1	.
g	.	1	.	1	.	.	.	1
h	.	.	1	.	1	.	.	.

The task solution is the search that contains 255 combinations. The minimum number of primitive rows that form the coverage is the optimal solution. The Hasse diagram [2-4] is compromise architecture with respect to time and memory. It leverages the previously obtained result to create a more complex superposition of solutions. For each coverage table containing n lines, it is necessary to generate an own Hasse sequencer for an almost parallel solution of the NP-complete problem. For instance, the four rows of the coverage table create the structure of the Hasse sequencer shown in Fig. 1.

The Hasse sequencer corresponds to the structural description of a closed alphabet formed by a Boolean on a universe of four primitives that represent all possible binary transitions of a logical variable in two automaton cycles [2]: B*(Y)={Q=(00), E=(01), H=(10), J=(11), O={Q, H}, I={E,J}, A={Q,E}, B={H,J}, S={Q,J}, P={E,H}, C={E,H,J}, F={Q,H,J}, L={Q,E,J}, V={Q,E,H}, Y={Q,E,H,J}, $U = \varnothing$.

The optimal solutions of the coverage problem for the matrix M, which generates 255 variants of possible combinations, are represented by rows in the DNF form: $C = fgh \vee efg \vee cdf$

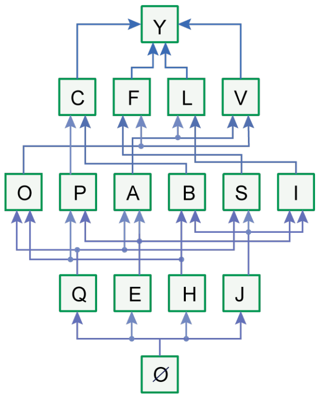


Fig. 1. Hasse-sequencer of the quantum computing

The control algorithm of the computational process for a quantum Hasse structure by an upward analysis of the graph vertices is based on the sequential execution of the following steps:

1. Entering the rows of the matrix in the first level registers $L_i^1 = P_i$ with the subsequent analysis of the coverage' quality where each matrix row (primitive) is evaluated by a bit: 1 means coverage presence, 0 means no coverage. If one of the primitives creates the coverage $\overset{m}{\underset{j=1}{\wedge}} L_{ij}^1 = 1$, the Hasse structure analysis ends. Otherwise, the transition (r = r + 1) to the next higher level of the graph is performed:

$$L_i^1 = P_i \rightarrow \overset{m}{\underset{j=1}{\wedge}} L_{ij}^1 = \begin{cases} 0 \rightarrow n = n+1; \\ 1 \rightarrow \text{end.} \end{cases}$$

2. Initiate a command for processing to the next level. Consecutive execution of vector (matrix) operations or, and: $L_i^r = L_{ij}^{r-1} \overset{m}{\underset{j=1}{\vee}} L_{tj}^{r-1}$, $\overset{m}{\underset{j=1}{\wedge}} L_{ij}^r = 1$ for analysis of the coverage, obtaining by superposition of r-level primitives. Here $t = \overline{1,m}, i = \overline{1,m}, r = \overline{1,n}$; n is the number of rows in the coverage table; m is the number of columns in it. If there is a superposition at the current level, creating a coverage indicator 1, the processing of all subsequent Hasse levels is not performed. Otherwise, the transition to the next higher level of the structure is performed:

$$L_i^r = L_{ij}^{r-1} \overset{m}{\underset{j=1}{\vee}} L_{tj}^{r-1} \rightarrow \overset{m}{\underset{j=1}{\wedge}} L_{ij}^r = \begin{cases} 0 \rightarrow r = r+1; \\ 1 \rightarrow \text{end.} \end{cases}$$

Every operational graph vertex consists of two register variables, which significantly reduces the hardware costs when implementing the Hasse sequencer. The number of clock cycles for processing the Hasse structure is at worst equal to n. It can also create an algorithm for search the optimal coverage by up-bottom analysis of the graph vertices. In this case, when the complete coverage in the current level is found, another descent along the structure is necessary to ensure that there is no complete coverage on the lower adjacent level. In this case, the solution obtained is optimal. Otherwise, it is necessary to perform the descent to a level where, the more lower, adjacent level will not contain a complete coverage. The vertices of the processor structure can have more than one register logical operation. Then the simplest command decoder to activate the logical operations: and, or, xor, not, is created.

Thus, the advantages of a quantum Hasse sequencer are the leverage of two-input elements for vector logical operations (and, or, xor), which makes it possible to significantly reduce hardware costs through the use of serial-parallel computations and a slight increase in the processing time of all the graph vertices. For each vertex, the coverage criterion is calculated, as the presence of all 1-units in the coordinates of the result vector. If the criterion is 1, then all other calculations are not performed, since the Hasse sequencer is a strictly hierarchical structure with respect to the number of superposition in each tier. This means that the best solution is at the lower level of the graph hierarchy. Variants of the same level are equivalent in cost of implementation; therefore the first coverage obtained ($Q = \sum_{i=1}^{n} q_i = n$) is the best solution, which implies stopping the Hasse-structure processing algorithm. The serial-parallel analysis cycle of the Hasse graph vertices is determined by the number of levels of the hierarchy or by the number of primitive rows in the coverage table multiplied by the time of analysis of one vertex: $T = \log_2 2^n \times t = t \times n$. In this case, the length m of the row of the coverage table does not affect the speed. Vertex analysis includes two instructions: logical (and, or, xor) and the operation of calculating the coverage criterion in the form of a scalar by applying the and-operation to all bits of the result vector:

$$m_{ir,j} = M_{i,j} \vee M_{r,j}, (j = \overline{1,n}; \{i \neq r\} = \overline{1,m};);$$

$$m^s_{ir} = \wedge m_{ir,j} = \wedge(M_{i,j} \vee M_{r,j})$$

Hardware costs for the implementation of the Hasse sequencer depend on the total number of graph vertices and on the number of bits in the row of the coverage table: $H = 2^n \times k \times m$, where k is the hardware implementation parameter of the binary vector logic operation and the criteria for calculating the quality of coverage.

Thus, the high speed of solving the coverage problem is achieved by significant increase hardware in $2^n \times k \times m / k \times m \times n = 2^n / n$ times, in comparison with the sequential processing of graph vertices. The Hasse sequencer provides the optimum between a completely parallel structure of computing processes, where the hardware cost is determined by the number of primitives at each node $H = k \times m \times n \times 2^n$, and the purely sequential computation structure, where the processing time of the Hasse graph is equal $T = t \times 2^n$ with minimal hardware costs $H^* = k \times m \times n$. Reducing the hardware complexity of the Hasse sequencer in comparison with the parallel processing of the graph is $Q^H = k \times m \times n \times 2^n / k \times m \times 2^n = n$. Reducing the time of analysis of the Hasse structure vertices, due to hardware redundancy, compared with a purely sequential processing of the graph vertices has the following estimate:

$$Q^T = \frac{t \times 2^n}{t \times n} = \frac{2^n}{n}.$$

The hardware-oriented Hasse architecture of the parallel computation of Boolean on the universum of n primitive rows is designed to solve the coverage problems, to minimize logic functions, to compress data, to synthesize and analyze digital systems. A prototype of a quantum Hasse structure implemented in software for the optimal solution of the coverage problem is used for the test minimization.

3. Method for synthesis a quasi-optimum test

The method is an integral part of the minimal test synthesis of digital circuits. It is based on the leverage of register parallel operations on the hardware-oriented data structures, which represent a qubit derivative matrix for black-box functionality. The architecture of the matrix sequencer for implementing the method is represented by the components shown in Fig. 2.

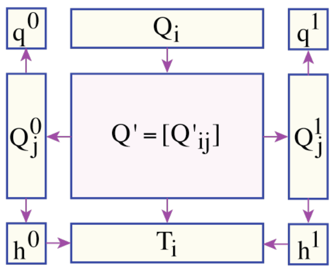


Fig. 2. The sequencer architecture for the quasi-optimal test synthesis

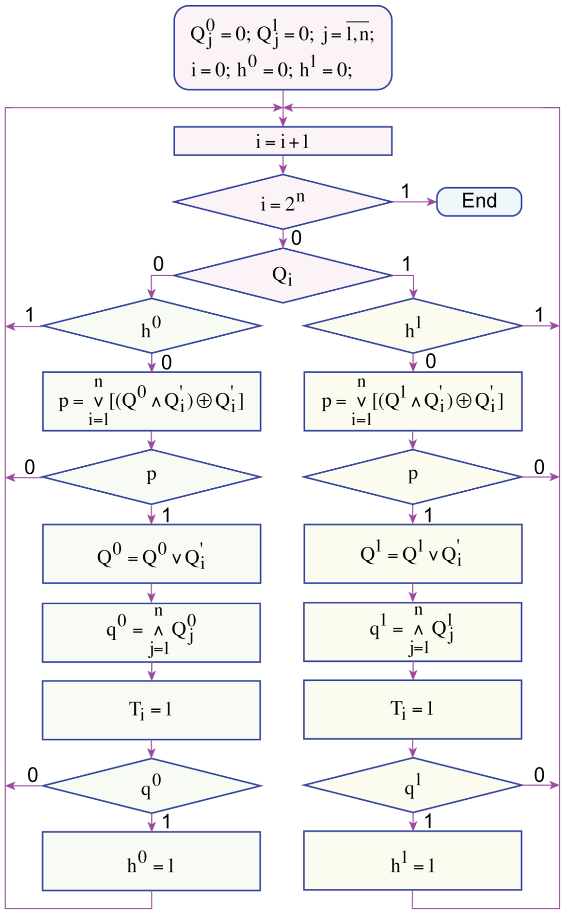
The analytical model and computational procedures for the test minimization using the qubit coverage matrix are the following:

$$A = <Q, Q', Q^0, Q^1, T, q^0, q^1, h^0, h^1, p>,$$
$$1)\ Q = (Q_1, Q_2, ..., Q_i, ..., Q_{2^n});$$
$$2)\ Q' = [Q'_{ij}], i = \overline{1, 2^n}, j = \overline{1, n};$$
$$3)\ Q^0, = (Q_1^0, Q_2^0, ..., Q_j^0, ..., Q_n^0);$$
$$4)\ Q^1, = (Q_1^1, Q_2^1, ..., Q_j^1, ..., Q_n^1);$$
$$5)\ T = (T_1, T_2, ..., T_i, ..., T_{2^n});$$
$$6)\ q^0 = (Q_1^0 \wedge Q_2^0 \wedge ... \wedge Q_j^0 \wedge ... \wedge Q_n^0);$$
$$7)\ q^1 = (Q_1^1 \wedge Q_2^1 \wedge ... \wedge Q_j^1 \wedge ... \wedge Q_n^1);$$
$$8)\ h^0 = 1 \leftrightarrow q^0 = 1;$$
$$9)\ h^1 = 1 \leftrightarrow q^1 = 1;$$
$$10)\ p = \bigvee_{j=1}^{n} [(Q_j^{0(1)} \wedge Q'_{ji}) \oplus Q'_{ji}].$$

Here are presented: 1) Qubit coverage of the black box functionality; 2) The matrix of qubit derivatives with respect to all variables; 3) Buffer accumulation register for indicating the process of obtaining a quasi-optimum coverage with respect to the zero Q-vector coordinates; 4) Buffer accumulation register for indicating the process of obtaining a quasi-optimum coverage by 1-unit coordinates of the Q-vector; 5) A qubit test vector, where test vectors are marked with 1-unit coordinates, which must be submitted to the Unit Under Test; 6) Indicator of the achievement of a quasi-optimum coverage with respect to the zero coordinates of the Q-vector; 7) Indicator of the achievement of a quasi-optimal coverage with respect to the 1-unit coordinates of the Q-vector; 8) The switch of the column analysis of the derivative matrix with respect to the zero coordinates of the Q-vector; 9) The switch of the column analysis of the derivative matrix with respect to the 1-unit coordinates of the Q-vector; 10) The coverage degree index of rows of the qubit derivatives matrix for the column under consideration. If the pointer is zero (there is no increment), then the column from the {0,1}-subsets of the matrix is not included in the test:

$$p = \bigvee_{i=1}^{n} [(Q^{0(1)} \wedge Q'_i) \oplus Q'_i].$$

The structural scheme of the quasi-optimal test synthesis algorithm based on splitting the matrix of derivatives has two symmetric branches, oriented to the analysis of the {0,1}-subsets of columns, respectively, Fig. 3. The basic idea of obtaining a quasi-optimal test is to find the minimum number of columns in the {0,1}-subsets of a qubit derivative matrix, cover all the rows or functionality variables with their 1-unit coordinates. In this case, if the next column does not add detection properties to the vectors previously included in the test, then it is excluded from the list.



$Q_j^0 = 0;\ Q_j^1 = 0;\ j = \overline{1, n};$
$i = 0;\ h^0 = 0;\ h^1 = 0;$
$i = i + 1$
$i = 2^n$
1
End
0
Q_i
h^0
h^1
$p = \bigvee_{i=1}^{n} [(Q^0 \wedge Q'_i) \oplus Q'_i]$
$p = \bigvee_{i=1}^{n} [(Q^1 \wedge Q'_i) \oplus Q'_i]$
p
$Q^0 = Q^0 \vee Q'_i$
$Q^1 = Q^1 \vee Q'_i$
$q^0 = \bigwedge_{j=1}^{n} Q_j^0$
$q^1 = \bigwedge_{j=1}^{n} Q_j^1$
$T_i = 1$
q^0
q^1
$h^0 = 1$
$h^1 = 1$

Fig. 3. Quasi-optimal test algorithm

The next step is the leverage of the described procedures for the quasi-minimal test synthesis of the Schneider logical example, presented in Fig. 4.

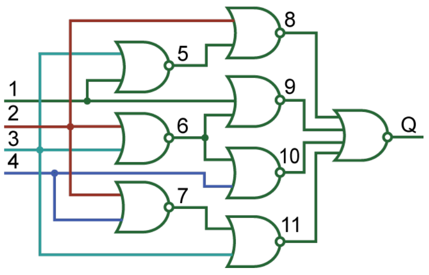


Fig. 4. Schneider logic circuit for the test synthesis

The qubit coverage of the digital circuit is represented by a vector (100000000000001), over which four qubit derivatives were obtained. Essentially, that for each variable the qubit derivative is composed of pairs of 1-unit coordinates in the vector. In aggregate, each pair of performing or-operation gives a functional term of three input variables, where there is no fourth line on which the derivative is taken. For instance, the derivative $Q'(X_1)=11$ at the address coordinates of the input variables 0000 and 1000 means the activation conditions of 000 for the first variable. Thus, the qubit derivative, as a pair of input vectors, is a test for single stuck-at-faults detection of the input variable under consideration. Naturally, there can be several pairs for each input line. In this case, their leverage is related to detecting the internal variables of the functionality. The following table shows the qubit coverage, Boolean qubit derivatives, and the states of the input variables that correspond to the values of the qubit coverage coordinates (for the good visual perception of the data picture, the zero states of the truth table and the derivative matrix coordinates are marked by points):

X_1	.	.	.	.	.	.	.	.	1	1	1	1	1	1	1	1
X_2	.	.	.	.	1	1	1	1	.	.	.	.	1	1	1	1
X_3	.	.	1	1	.	.	1	1	.	.	1	1	.	.	1	1
X_4	.	1	.	1	.	1	.	1	.	1	.	1	.	1	.	1
Q	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
$Q'(X_1)$	1	.	.	.	.	.	.	1	1	.	.	.	.	.	.	1
$Q'(X_2)$	1	.	.	.	1	.	.	.	.	.	.	1	.	.	.	1
$Q'(X_3)$	1	.	1	.	.	.	.	.	.	.	.	.	.	1	.	1
$Q'(X_4)$	1	1	.	.	.	.	.	.	.	.	.	.	.	.	1	1
Test =	1	1	1	.	1	.	.	1	1	.	.	1	.	1	1	1
T_{min} =	1	1	.	.	.	.	.	1	1	.	.	.	.	.	1	1

The result of performing the or-operation on the qubit derivative vectors contains a test with 10 input vectors T = (1110100110010111). In general, the matrix of qubit derivatives with its 1-unit values creates a complete, but overhead test for detection all single stuck-at-faults of external and internal variables. Since it, by pairing input vectors, activates all logical paths from the input lines to the outputs of the digital black box functionality. Sensitivity, as a property of digital functionality, is the ratio of the number of 1-unit coordinates in the qubit derivative matrix to the total number of coordinates. For instance, the sensitivity of the Schneider circuit is 0.25.

The procedure for the test minimization, taking into account the structure of the circuit, by finding the quasi-minimal coverage of all input variables by {0,1}-subsets of the qubit derivatives, generates the six test vectors represented in the following table (the points indicate the coordinates with undetected faults):

Test	1	2	3	4	5	6	7	8	9	10	11	12	FD	FC
000011100001	1	1	1	1	0	0	0	1	1	1	1	0	50	50
000111000010	.	.	.	0	.	.	1	.	.	.	0	1	16	66
011100001000	.	.	.	.	.	1	.	.	0	.	.	1	12	75
100001110000	0	.	.	.	1	.	.	0	.	.	.	1	16	87
111000000100	.	.	.	.	.	1	.	.	.	0	.	1	12	91
111100000001	0	0	0	0	.	.	.	1	1	1	1	0	37	100
$\cup$ =	X	X	X	X	X	X	X	X	X	X	X	X		

Thus, the minimal test for the Schneider logic contains only six input sequences that detect 100 percent of single stuck-at-faults for the input, internal and output lines of the device. The density of the faults detected on a given test is defined as the ratio of the defects being detected to the total number of the fault detection table coordinates. For the Schneider circuit, the test has a fault density of 35/72 = 0.49.

The experiments performed to minimize tests on 16 fragments of digital devices (4-10 input variables) indicate the following: 1) In 25 percent of cases optimal tests were obtained. 2) In 70 percent of the cases, the tests differed from the optimal ones by no more than 25 percent. 3) In 5 percent of the cases, the tests were close to the exhaustive number of input sequences. 4) The computational complexity of the proposed qubit method for synthesis the minimum test for logical functionality from n variables is determined by the estimate forming the time costs for taking the qubit derivatives and the test minimization:

$$E = 2n + 2n^2 = 2n(n+1).$$

Thus, the method of test synthesis based on the leverage qubit derivatives allows the generation of input vectors detecting all single stuck-at-faults of the input and internal lines. However, for the synthesis of the minimum test, it is necessary to use the structure of a digital device.

4. Conclusion

The innovation of the results is as follows:

1) The Hasse sequencer, focused on the parallel solution of quasi-optimal search problems, in particular to minimize the test, detecting single stuck-at-faults in digital black-box functionality.

2) The method, algorithm and sequencer structure for synthesis and minimization tests of black-box functionality is proposed, using qubit derivative matrix to find the quasi-optimum coverage. 3) Experiments were carried out on the fragments of digital circuits, which showed, the practical significance and high performance of the proposed architecture and the method of synthesis a quasi-minimal test for black-box logical functionality. 4) Further research will be directed to the creation a family of intelligent algorithms for the test synthesis, simulation and fault diagnosis leveraging the qubit coverage and derivatives apparatus.

10. References

[1] Hiroshi I. and Masahito H. (2006) Quantum Computation and Information. From Theory to Experiment. Springer.

[2] Hahanov V.I., Hahanova I.V., Litvinova E.I., and Guz O.A. (2010) Design and verification of SoC. Verilog & System Verilog. Kharkov. Novoe Slovo.

[3] Hahanov V.I., Litvinova E.I., Chumachenko S.V., and Guz O.A. (2011) Logical associative processor. Electronic modeling. J 1: 73-90.

[4] Hahanov V.I. Hahanova I.V., Guz O.A., and Abbas M.A. (2012) Quantum models for data structures and computing. In: TCSET Proceedings. Lvov. Slavske.

[5] V. Hahanov et al., "Qubit test synthesis of the functionality," 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, 2017, pp. 251-255.